Meta Bayesian Optimization to Discover a Problem Worth Optimizing

Yuki Takezawa [1] **Masaki Adachi** [1]
[1]*Lattice Lab, Toyota Motor Corporation*. Correspondence to: Masaki Adachi .

1. Introduction

Bayesian optimization (BO) is a sample-efficient black box optimizer and has become a workhorse in self-driving labs. Yet a major and often overlooked source of uncertainty is the *task itself*: *what* to optimize, *how* to evaluate success, and *which* design space and constraints define a meaningful scientific problem. Humans still make this problem-selection decision; automating it would let labs explore beyond human-designed objectives.

We call a candidate problem setting—its search space, objective, and constraints—a *task*. A task is valuable only if it is both *high-reward* and *achievable* within a finite experimental budget T: overly ambitious tasks may not mature in time, while only pursuing easy tasks can limit impact. Large language models (LLMs) make it natural to envision *open-ended task exploration*, where algorithms continually propose and revise tasks. However, two obstacles remain: (i) evidence-based discovery loops are brittle due to hallucinations [1]; (ii) sample efficiency, since testing all generated tasks is prohibitive.

In response, we propose **Generate–Select–Refine (GSR)**, a meta-BO framework that alternates between *task generation* and *task optimization* (Fig. 1). The key is to treat task discovery as a *meta-level optimization problem*, using BO regret bounds as a statistical *task manager* that verifies generated tasks from finite evidence. Asymptotically, GSR concentrates evaluations on the best task, incurring only a regret overhead logarithmically away from running single-task BO.

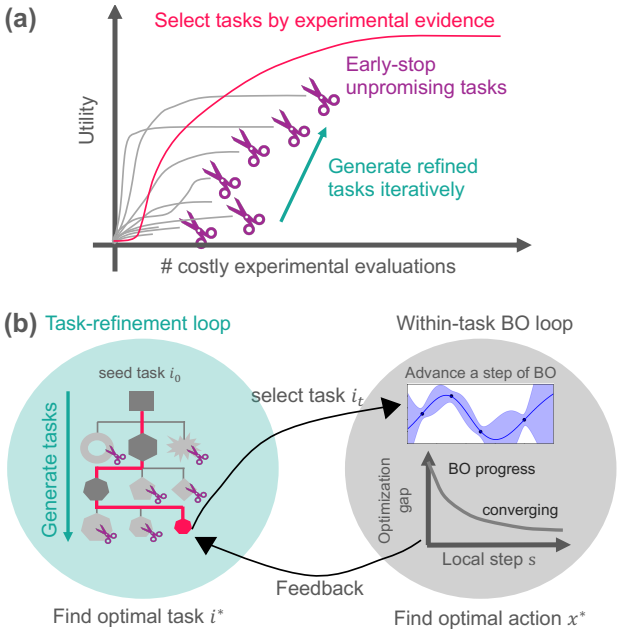

Fig. 1: Conceptual overview: starting from a seed task, GSR iteratively refines task specifications while allocating costly evaluations. (a) As evidence accumulates, the selector focuses on promising tasks and early-stops unpromising ones. (b) Two-loop structure: within-task BO provides progress signals; the meta-level uses utilities and BO "optimization gap" to decide what to optimize next.

2. GSR: Meta-BO for task discovery

2.1 Two coupled loops: inner loop vs. outer loop

Inner loop (standard BO). For each task i, we optimize an unknown black-box objective $f^{(i)}(x)$ over its domain $\mathcal{X}^{(i)}$ (e.g., via GP-UCB [2]). This produces an *incumbent* $z_s^{(i)}$, the best value found so far after s evaluations.

Outer loop (task selection + task refinement). Tasks can differ in domains, objectives, and even dimensionalities, so raw $f^{(i)}$ values are not directly comparable. We therefore compare tasks through a bounded *utility* $u^{(i)}(z) \in [0,1]$ applied to each task's incumbent. Utility can encode scientific promise, deployability, or expert preference; in practice it can come from human experts or an LLM committee.

2.2 Core mechanism: meta-confidence interval

Early incumbents can be misleading: a task may look good simply because it has not been explored enough. GSR addresses this with **meta-confidence intervals** (CIs): optimistic and pessimistic bounds on each task's *long-run* value $U^{(i)} := u^{(i)}(f^{\star(i)})$, where $f^{\star(i)}$ is the (unknown) optimum of task i.

Concretely, GSR combines: (i) statistical uncertainty in utility feedback, and (ii) *optimization uncertainty* within the task (how far the incumbent may still be from the true optimum under a finite budget). This yields a meta-UCB style upper bound of the form

$$\underbrace{V^{(i)}}_{\text{eventual task value}} \lesssim \underbrace{\mathrm{UCB}_u^{(i)}(s)}_{\text{utility uncertainty}} + \underbrace{\bar{L}\,\epsilon_f^{(i)}(s)}_{\text{BO optimization gap}}, \tag{1}$$

where $\epsilon_f^{(i)}(s)$ is the standard GP-UCB optimization gap term after s evaluations and $\bar{L}$ is a meta-level Lipschitz bound translating objective improvement into utility improvement. Intuitively, tasks with mediocre incumbents but large remaining optimization gap should not be discarded too early; conversely, tasks with little optimization gap may have saturated.

2.3 Generate, Select, Refine

GSR uses the following three modules and LLM generator and evaluator (Figure 2).

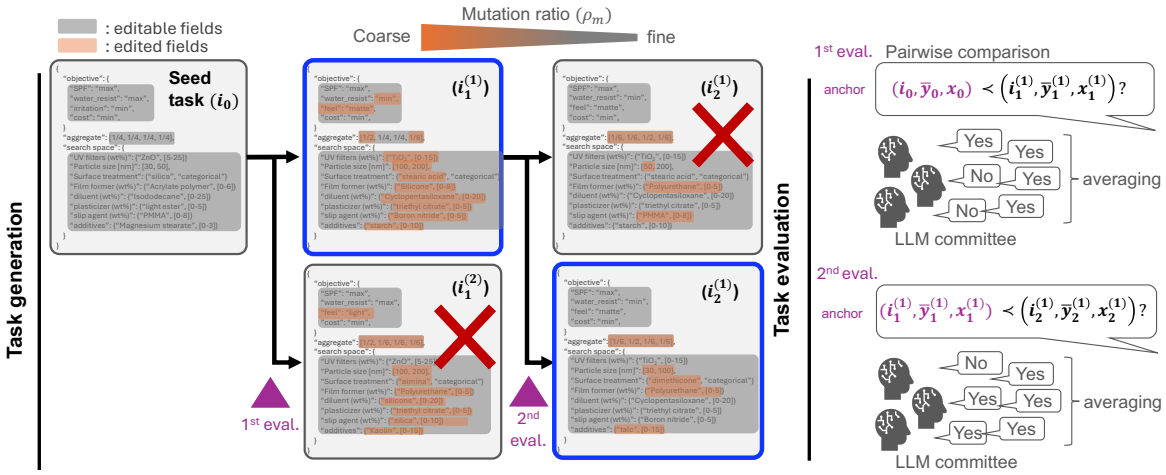

Fig. 2: LLM components in GSR: (**Left**) Coarse-to-fine task generation: we mutate a parent task with target mutation ratio ρ_m, producing refined tasks. (**Right**) Committee-based evaluation: committee returns binary votes if the current task–incumbent pair $(i, \bar{y}_s^{(i)})$ is preferred over an anchor $(a_t, \bar{y}_s^{(a_t)})$. By averaging them, we estimate a confidence interval for the utility $u^{(i)}(\bar{y}_s^{(i)})$.

Generate	Propose refined tasks by mutating a parent task (e.g., via an evolutionary LLM).
Select	Allocate the next evaluation to the task with meta-UCB, while early-stopping tasks that provably cannot catch up.
Refine	Only refine a promising task using a coarse-to-fine mutation schedule.

2.4 What GSR guarantees (high level)

For applied audiences, the main takeaway is: **GSR is provably sample-efficient for open-ended task discovery.** Under standard GP-UCB conditions and mild assumptions on the task generator, GSR's cumulative meta-regret is only logarithmically worse than running BO on a single (best) task. Practically, this means GSR can explore an open-ended set of generated tasks without incurring prohibitive experimental cost, while still concentrating evaluations on the best task w.h.p. To our knowledge, this is the first such regret guarantee for open-ended task discovery settings, including those driven by evolutionary LLM-based generators such as AlphaEvolve [3].

2.5 Related Work

Our setting differs from multi-objective and multi-task BO: rather than optimizing multiple objectives over a shared space or transferring across related tasks, we aim to *discover the task itself* (domain/objective/constraints).

2.6 Applications and empirical evidence

GSR is particularly useful for:

(a) **Planning:** discovering a task formulation that makes a desired outcome attainable under realistic budgets (e.g., product or process planning).

(b) **Inverse optimization:** given a discovered material, searching for a task under which it becomes optimal—i.e., a principled route to finding applications of the new material.

We also demonstrate **patent repurposing**: given a "seed" material that did not fit its original target application, we search for alternative tasks where the same material becomes optimal (competitive over rival materials). In the instantiation, using MgO and Bi_2O_3 as seeds, GSR identifies plausible additives (e.g., $TmCl_3/HgF_2$ for MgO and $CuO/PrTe_2$ for Bi_2O_3) and proposes repurposed applications, illustrating *data-supported* hypothesis generation rather than hypotheses derived solely from an LLM.

2.7 Practical takeaway for AI4X practitioners

To deploy GSR in an applied pipeline, you need:

(1) **A seed task:** an initial task formulation (possibly vague or imperfect).

(2) **A utility function:** any mechanism that can rank task–incumbent pairs (human experts, automated metrics, or LLM-as-a-judge).

If LLM reliability is a concern, both the generator and the utility oracle can be replaced by humans, and utility queries can be made sparse (queried intermittently) while retaining theoretical guarantees up to logarithmic factors.

3. Conclusion

BO is powerful once the task is fixed, but in many real discovery settings **the task itself must be discovered under budget constraints**. GSR provides a unified, evidence-based framework for LLM-assisted (or human-assisted) task discovery, with regret guarantees that justify exploring many candidate tasks without losing sample efficiency. We view this as a step toward self-driving laboratories that not only optimize experiments, but also help decide *what to optimize next*. See Appendix for formal details.

References

[1] Zhangde Song, Jieyu Lu, Yuanqi Du, Botao Yu, Thomas M Pruyn, Yue Huang, Kehan Guo, Xi-

uzhe Luo, Yuanhao Qu, Yi Qu, et al. Evaluating large language models in scientific discovery. *arXiv preprint arXiv:2512.15567*, 2025.

[2] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*, pages 1015–1022, 2010.

[3] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[4] Madison Lee and Tara Javidi. Consequences of kernel regularity for bandit optimization. *arXiv preprint arXiv:2512.05957*, 2025.

[5] Raphael D Levine. *Molecular reaction dynamics*. Cambridge university press, 2009.

[6] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[7] Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

[8] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[9] Masahiro Fujisawa, Masaki Adachi, and Michael A Osborne. Scalable valuation of human feedback through provably robust model alignment. *arXiv preprint arXiv:2505.17859*, 2025.

[10] Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or LLMs as the judge? a study on judgement bias. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8301–8327, 2024.

[11] Chungpa Lee, Thomas Zeng, Jongwon Jeong, Jy-yong Sohn, and Kangwook Lee. How to correctly report LLM-as-a-judge evaluations. *arXiv preprint arXiv:2511.21140*, 2025.

[12] Juliusz Ziomek, Masaki Adachi, and Michael A Osborne. Bayesian optimisation with unknown hyperparameters: regret bounds logarithmically closer to optimal. *Advances in Neural Information Processing Systems*, 37:86346–86374, 2024.

[13] Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. CodeEvolve: An open source evolutionary coding agent for algorithm discovery and optimization. *arXiv preprint arXiv:2510.14150*, 2025.

[14] Jingfeng Wu, Difan Zou, Zixiang Chen, Vladimir Braverman, Quanquan Gu, and Peter Bartlett. How many pretraining tasks are needed for in-context learning of linear regression? In *The Twelfth International Conference on Learning Representations*, 2024.

[15] Tomoya Wakayama and Taiji Suzuki. In-context learning is provably bayesian inference: a generalization theory for meta-learning. *arXiv preprint arXiv:2510.10981*, 2025.

[16] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[17] Michael Volpp, Lukas P. Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for transfer learning in bayesian optimization. In *International Conference on Learning Representations*, 2020.

[18] Zhongxiang Dai, Yizhou Chen, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. On provably robust meta-Bayesian optimization. In *Uncertainty in Artificial Intelligence (UAI)*, pages 475–485. PMLR, 2022.

[19] Alexandre Maraval, Matthieu Zimmer, Antoine Grosnit, and Haitham Bou Ammar. End-to-end meta-bayesian optimisation with transformer neural processes. *Advances in Neural Information Processing Systems*, 36:11246–11260, 2023.

[20] Zi Wang, Beomjoon Kim, and Leslie P Kaelbling. Regret bounds for meta bayesian optimization with an unknown gaussian process prior. *Advances in Neural Information Processing Systems*, 31, 2018.

[21] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. No-regret bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research*, 20(50):1–24, 2019.

[22] Juliusz Ziomek, Masaki Adachi, and Michael A Osborne. Time-varying gaussian process bandits with unknown prior. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258, pages 4294–4302, 2025.

[23] Zonglin Yang, Xinya Du, Junxian Li, Jie Zheng, Soujanya Poria, and Erik Cambria. Large language models for automated open-domain scientific hypotheses discovery. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13545–13565, 2024.

[24] Santiago Miret and NM Anoop Krishnan. Enabling large language models for real-world materials discovery. *Nature Machine Intelligence*, 7(7):991–998, 2025.

[25] Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (stop): Recursively self-improving code generation. In *First Conference on Language Modeling*, 2024.

[26] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

[27] Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. In *The Twelfth International Conference on Learning Representations*, 2024.

[28] James Requeima, John Bronskill, Dami Choi, Richard E. Turner, and David Duvenaud. Llm processes: Numerical predictive distributions conditioned on natural language. In *Advances in Neural Information Processing Systems*, volume 37, pages 109609–109671, 2024.

[29] Rushil Gupta, Jason Hartford, and Bang Liu. LLMs for Bayesian optimization in scientific domains: Are we there yet? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 15482–15510. Association for Computational Linguistics, 2025.

[30] Masaki Adachi, Brady Planden, David Howey, Michael A. Osborne, Sebastian Orbell, Natalia Ares, Krikamol Muandet, and Siu Lun Chau. Looping in the human: Collaborative and explainable Bayesian optimization. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238, pages 505–513, 2024.

[31] Wenjie Xu, Masaki Adachi, Colin N. Jones, and Michael A. Osborne. Principled bayesian optimization in collaboration with human experts. In *Advances in Neural Information Processing Systems*, volume 37, pages 104091–104137. Curran Associates, Inc., 2024.

[32] Richard Cornelius Suwandi, Feng Yin, Juntao Wang, Renjie Li, Tsung-Hui Chang, and Sergios Theodoridis. Adaptive kernel design for bayesian optimization is a piece of cake with llms. *arXiv preprint arXiv:2509.17998*, 2025.

[33] Huong Ha, Santu Rana, Sunil Gupta, Thanh Nguyen, Svetha Venkatesh, et al. Bayesian optimization with unknown search space. *Advances in Neural Information Processing Systems*, 32, 2019.

[34] Sunil Gupta, Santu Rana, Huong Ha, Svetha Venkatesh, et al. Sub-linear regret bounds for bayesian optimisation in unknown search spaces. *Advances in Neural Information Processing Systems*, 33:16271–16281, 2020.

[35] Majid Abdolshah, Alistair Shilton, Santu Rana, Sunil Gupta, and Svetha Venkatesh. Multi-objective bayesian optimisation with preferences over objectives. *Advances in neural information processing systems*, 32, 2019.

[36] Zhiyuan Jerry Lin, Raul Astudillo, Peter Frazier, and Eytan Bakshy. Preference exploration for efficient bayesian optimization with multiple outcomes. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151, pages 4235–4258, 2022.

Appendix A. Problem Setting

1.1 Tasks, evaluations, and incumbents

Tasks. There is an unbounded collection of tasks indexed by $i \in \mathbb{N}$. Task i is specified by a compact domain $\mathcal{X}^{(i)} \subset \mathbb{R}^{d^{(i)}}$ and an unknown objective $f^{(i)} : \mathcal{X}^{(i)} \to \mathbb{R}$.

Global vs. local time. We index global rounds by $t \in [T]$. At each t, the meta-optimizer selects a task i_t and evaluates one design $x_t^{(i_t)} \in \mathcal{X}^{(i_t)}$, observing the noisy oracle

$$y_t^{(i_t)} = f^{(i_t)}\big(x_t^{(i_t)}\big) + \xi_t^f, \qquad \xi_t^f \sim \mathrm{SubGauss}(\sigma_f^2).$$

Let $s_t^{(i)}$ be the local counter that task i has been selected until t and $\mathcal{I}_T$ be the set of tasks instantiated by T. For brevity, we write $x_t^{(i)} := x^{(i)}_{s_t^{(i)}}$ (and similarly for other quantities).

Incumbent. After s local evaluations on task i, define the incumbent, i.e., best-so-far observed value[1]

$$\overline{y}_s^{(i)} := \max_{1\le r\le s} y_r^{(i)}, \qquad f^{\star(i)} := \max_{x\in\mathcal{X}^{(i)}} f^{(i)}(x).$$

Optimization gap. Each task runs a standard GP-based BO (GP-UCB; [2]). We only use an *optimization gap bound*: w.p. $\ge 1-\delta_f^{(i)}$, after s evaluations,

$$f^{\star(i)} - \overline{y}_s^{(i)} \;\le\; \epsilon_s^f := 2\sqrt{C_\lambda\, \beta_s^{(i)}\, \gamma_s^{(i)}/s}, \qquad \text{(A1)}$$

where $\beta_s^{(i)}$ and $\gamma_s^{(i)}$ are exploration and information-gain terms and $C_\lambda > 0$ is a constant. The optimization gap ϵ_s^f upper-bounds the worst-case error of the global optimum estimate, and satisfies $\epsilon_s^f \to 0$ as $s \to \infty$, i.e., GP-UCB asymptotically identifies the global optimum $f^{\star(i)}$ (no-regret) for commonly used kernels (see [4]). ϵ_s^f is computable without access to $f^{\star(i)}$, so it provides a principled cross-task progress metric. For uniformity over instantiated tasks, define $\overline{\epsilon}_s^f := \max_{i\in\mathcal{I}_T} \epsilon_s^f$, so $\epsilon_s^f \le \overline{\epsilon}_s^f$ for all $i \in \mathcal{I}_T$ and $s \in [T]$.

1.2 Latent utility

Since the scale of $f^{(i)}$ can vary across tasks, we introduce a scale-invariant utility function $u^{(i)} : \mathbb{R} \to [0,1]$ applied to the incumbent $\overline{y}_s^{(i)}$. Any rankable function can be utility; e.g., scientific value [5] or human preference [6].

Definition A.1 (Utility regularity)**.** $u^{(i)}$ is monotone nondecreasing and $L^{(i)}$-Lipschitz: $|u^{(i)}(y) - u^{(i)}(y')| \le L^{(i)}|y-y'|$ for all $y, y' \in \mathbb{R}$. Let $L^\star := \sup_i L^{(i)} < \infty$.

Monotonicity matches the optimization intent (better designs should not be judged worse). Lipschitzness is a standard regularity condition in optimization.

Noisy utility calls. For task i_t selected at t, after observing $y_t^{(i_t)}$, we query the noisy utility oracle

$$\tilde{u}_t^{(i_t)} = u^{(i_t)}\big(\overline{y}_t^{(i_t)}\big) + \xi_t^u, \qquad \xi_t^u \sim \mathrm{SubGauss}\Big(\sigma_u^2\Big).$$

[1] In analysis, this is noiseless $\overline{y}_s^{(i)} := \max_{1\le r\le s} f^{(i)}(x_r^{(i)})$.

1.3 Long-run value and BO-achievable optimality

Long-run value. A task i has an unknown long-run value

$$U^{(i)} := u^{(i)}\big(f^{\star(i)}\big) \in [0,1], \qquad U^\star := \sup_i U^{(i)}.$$

Resolution. With a finite T, the optimization gap is nonzero $\epsilon_T^f > 0$. Combining Lipschitzness with (A1) gives, $\forall i, s$

$$0 \le U^{(i)} - u^{(i)}(\overline{y}_s^{(i)}) \le L^{(i)}\big(f^{\star(i)} - \overline{y}_s^{(i)}\big) \le L^\star\, \epsilon_s^f,$$

Consequently, even if all budget T is allocated to a single task, $L^\star \epsilon_T^f$ is the finest utility accuracy that can be reliably achieved. We therefore define an *achievable* utility tolerance (the "resolution").

Definition A.2 (ϵ^U-optimal tasks)**.** For any $\epsilon^U > 0$, define

$$\mathcal{I}^\star(\epsilon^U) := \big\{ i \in \mathbb{N} :\; U^\star - U^{(i)} \le \epsilon^U \big\}.$$

Condition A.3 (Optimal resolution)**.** *Fix $\epsilon_m^U := \epsilon_0^U 2^{-m}$ for $m \in \mathbb{Z}_{\ge 0}$. Choose a max depth $\overline{m}_T \in \mathbb{Z}_{\ge 0}$ and assume there exists an $m^\star \in \{0, 1, \dots, \overline{m}_T\}$ and $c_\epsilon \in (0,1)$ s.t.*

$$\epsilon_{m^\star}^U \;\le\; c_\epsilon\, L^\star\, \overline{\epsilon}_T^f$$

Condition A.3 ensures that the $\epsilon_{m^\star}^U$ is *achievable* within T; otherwise, any finer resolution $\epsilon_{\overline{m}_T}^U < \epsilon_{m^\star}^U$ would exceed what BO can reliably achieve. As T increases, $\overline{\epsilon}_T^f$ shrinks and $\epsilon_{m^\star}^U$ becomes progressively finer.

Goal is to find and optimize any $\epsilon_{m^\star}^U$-optimal task:

$$i^\star_{\epsilon^U} \in \mathcal{I}^\star(\epsilon_{m^\star}^U), \qquad x^{\star(i^\star_{\epsilon^U})} \in \arg\max_{x\in\mathcal{X}^{(i^\star_{\epsilon^U})}} f^{(i^\star_{\epsilon^U})}(x). \quad \text{(A2)}$$

Metric. We evaluate the meta-optimizer by the meta regret:

$$\mathcal{R}_T := \sum_{t=1}^{T} \Big(U^\star - u^{(i_t)}\big(\overline{y}_t^{(i_t)}\big) \Big).$$

Appendix B. Generate-Select-Refine (GSR)

We propose GSR (Alg. 1), a meta-BO framework for *online task discovery* (generate), *budget allocation* (select), and *coarse-to-fine scheduler* (refine). This section is intentionally *algorithm-first*: we define the quantities GSR maintains and explain the control flow. Concrete LLM instantiations of the generator and utility oracle are given in §B.2. All formal guarantees are collected in §C.

2.1 Two coupled loops: GP-UCB and meta-UCB

Each task i runs a standard GP-based BO procedure (GP-UCB), maintaining: (i) an incumbent $\overline{y}_s^{(i)}$ after s local evaluations, and (ii) a computable optimization gap ϵ_s^f from Eq.(A1). The meta-optimizer decides (a) which task to evaluate next (select) and (b) when to generate tasks (refine).

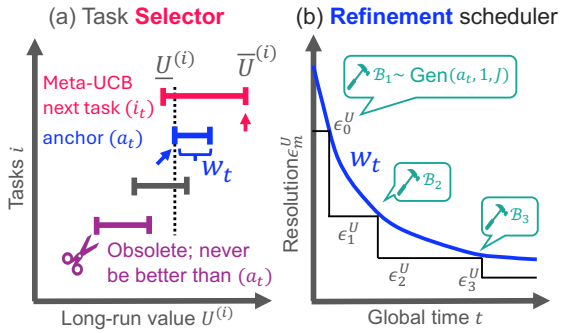

Fig. A1: Alg. 1: (a) **Select** chooses the next task i_t and anchor task a_t. (b) **Refine** advances the resolution levels ϵ_m^U and generates J mutations when anchor width $w_t \le c_g \epsilon_m^U$.

Task Selection. Core idea is the confidence intervals (CIs):

$$U^{(i)} \in [\underline{U}_s^{(i)}, \overline{U}_s^{(i)}],$$
$$\overline{U}_s^{(i)} = \underbrace{\overline{u}_s^{(i)}}_{\text{Utility UCB}} + \underbrace{\overline{L}\epsilon_s^f}_{\text{optimization gap}},$$

where $\overline{L} \ge L^\star$ bounds the Lipschitz constant. The UCB naturally decomposes into uncertainty from finite utility samples and an optimization gap due to limited evaluation budget. We select the next task i_t via a meta-UCB policy (Line 4 in Alg. 1; see Fig.A1(a))[2], concentrating budget on the ϵ^U-optimal task while early-stopping suboptimal ones.

Task Refinement. We generate new tasks by mutating and iteratively refining a seed task. A key step is selecting an *anchor* task to mutate. The anchor must be both promising and sufficiently resolved. We therefore choose a_t as the most promising pessimistic task—ranked by $\underline{U}_t^{(i)}$ rather than $\overline{U}_t^{(i)}$—whose uncertainty is below a target threshold:

$$a_t \in \arg\max_{j \in \mathcal{I}_t} \underline{U}_t^{(j)} \text{ s.t. } w_t^{(j)} \le \overline{\epsilon}_t^U, \tag{A3}$$

where $w_t^{(i)} := \overline{U}_t^{(i)} - \underline{U}_t^{(i)}$ is the CI width (uncertainty), and $\overline{\epsilon}_t^U := \max\{c_g \epsilon_m^U, \underline{w}_t\}$ is the resolution threshold, with $\underline{w}_t := \min_{j \in \mathcal{I}_t} w_t^{(j)}$. See Fig.A1(a) for intuition. This rule selects an anchor that is promising under $\underline{U}$ and sufficiently certain to advance to the next level m (cf. Condition A.3).

Generation scheduler. We generate new tasks only once the current anchor is sufficiently certain (Line 8; Fig.A1(b)). We then advance to m+1 and generate a new batch of child tasks at a finer mutation scale ϵ_m^U.

2.2 LLM-powered meta-BO

We use LLMs for both task generation and evaluation. Specifically, we adopt two empirically successful paradigms (Fig.2): evolutionary LLMs for task generation [3, 7] and preference-based LLM judges for utility feedback [8, 9]. We present only the essential details here.

[2]Ties are broken in favor of the task with fewer evaluations.

Algorithm 1 Generate-Select-Refine (GSR)

1: **Input:** horizon T, confidence $(\delta_f, \delta_u, \delta_-)$, Lipschitz bound $\overline{L}$, seed task i_0, batch size J, c_g, $\overline{m}_T$.
2: **Init:** $m \leftarrow 0$, $\epsilon_0^U \leftarrow 1$, $\epsilon_m^U \leftarrow \epsilon_0^U$, $\mathcal{I}_1 \leftarrow \{i_0\}$, $\mathcal{B}_0 \leftarrow$ Gen$(i_0, 0, J)$, $\mathcal{I}_1 \leftarrow \mathcal{I}_1 \cup \mathcal{B}_0$.
3: **for** $t \in [T]$
4: select the next task $i_t \in \arg\max_{j \in \mathcal{I}_t} \overline{U}_t^{(j)}$
5: $x_t^{(i_t)}, y_t^{(i_t)}, \overline{y}_t^{(i_t)} \leftarrow$ RunOneStepGP-UCB(i_t).
6: update value envelopes for task i_t by Theorem C.3.
7: select anchor a_t, w_t by Eq. (A3); $\mathcal{I}_{t+1} \leftarrow \mathcal{I}_t$.
8: **if** $m < \overline{m}_T$ **and** $w_t \le c_g \epsilon_m^U$
9: step up $m \leftarrow m + 1$; $\epsilon_m^U \leftarrow \epsilon_0^U 2^{-m}$.
10: draw $\mathcal{B}_m \leftarrow$ Gen(a_t, m, J); $\mathcal{I}_{t+1} \leftarrow \mathcal{I}_{t+1} \cup \mathcal{B}_m$.

2.3 Generate tasks by controlled mutations

We represent each task as a structured specification (e.g., JSON) with editable fields (e.g., objective/domain). Given an anchor task a_t and level m, the generator Gen(a_t, m, J) prompts an LLM to propose mutated child tasks $\mathcal{B}_m$.

Mutation ratio. To operationalize the resolution levels ϵ_m^U, we control the mutation ratio:

$$\rho(i, a_t) := \frac{\#\text{modified editable fields}}{\#\text{editable fields}} \in [0, 1].$$

We target a decreasing schedule $\rho_m := \rho_0 2^{-m}$, so higher m corresponds to smaller edits. In practice, we implement this by prompting the LLM with an explicit constraint on the number/type of fields (Fig.2).

2.4 Utility elicitation via an LLM committee

Following standard LLM-as-a-judge practice [10, 11], an LLM committee compares task pairs and returns binary votes (Fig.2). We adopt the Bradley–Terry model [6] to infer latent utilities from these comparisons, defining the utility $u^{(i)}(\cdot)$ as the win rate of task i_t against the seed task i_0, i.e., $\Pr(i_t \succ i_0 \mid \overline{y}_t^{(i_t)}, \overline{y}_0^{(i_0)}, x_t^{(i_t)}, x_0^{(i_0)})$. As both the task i_t and its solution $\overline{y}_t^{(i_t)}$ improve, this win rate provides a consistent cross-task utility. To prevent saturation when i_t is almost always preferred, we instead use an anchor task a_t as an adaptive reference and recover the win rate by chaining pairwise comparisons (e.g., $i_t \succ a_t \succ i_0$).

Remark B.1. Under an LLM committee, averaging K_t preference votes yields a single utility estimate $\tilde{u}_t \in [0, 1]$ with $1/(4K_t)$-sub-Gaussian noise, and the resulting win rate satisfies Definition A.1.

Appendix C. Theoretical Analysis

3.1 Main regret bound

Confidence envelopes. We now establish the regret bound for GSR. As a first step, we derive CIs for the meta-UCB.

Lemma C.1 (Utility CIs)**.** *Let $\delta_t := \delta_u/(\pi^2 t^2)$ and $\phi_t := \sqrt{2\sigma_u^2 \log(2/\delta_t)}$. Then w.p.$\geq 1-\delta_u$, for all $t \in [T]$,*

$$\underline{u}_t^{(i_t)} := clip_{[0,1]}(\tilde{u}_t^{(i_t)} - \phi_t),\ \overline{u}_t^{(i_t)} := clip_{[0,1]}(\tilde{u}_t^{(i_t)} + \phi_t),$$

Remark C.2 (Averaged feedback). Averaging $K_t \geq 1$ independent samples to form $\tilde{u}_t$ is (σ_u^2/K_t)-sub-Gaussian.

Theorem C.3 (Value envelopes)**.** *Under Lemma C.1 and the GP-UCB events* (A1)*, if $\overline{L} \geq L^\star$ then, $\forall i \in \mathcal{I}_t, t \leq T$,*

$$\underline{U}_t^{(i)} := \underline{u}_t^{(i)}, \quad \overline{U}_t^{(i)} := clip_{[0,1]}(\overline{u}_t^{(i)} + \overline{L}\,\epsilon_t^f),$$

with initialization $[\underline{U}_0^{(i)}, \overline{U}_0^{(i)}] = [0,1]$. If K_t is controlled to be $\overline{u}_t^{(i)} - \underline{u}_t^{(i)} \leq c_u \overline{L}\,\epsilon_t^f$ for $c_u > 0$, then

$$w_t^{(i)} := \overline{U}_t^{(i)} - \underline{U}_t^{(i)} \leq (c_u+1)\overline{L}\,\epsilon_t^f.$$

Task generation.

Condition C.4 (Refinement probability)**.** *Under Condition A.3, there exists a nonzero probability $\delta_+ \in (0,1]$ such that $\forall m, s, a_t$ that is resolved finer than ϵ_m^U, a single draw $i \sim \mathsf{Gen}(a_t, m, J=1)$ hits an ϵ_m^U-optimal task for $m \geq 1$:*

$$\Pr\Big(i \in \mathcal{I}^\star(\epsilon_m^U) \,\Big|\, w_t \leq c_g \epsilon_{m-1}^U\Big) \geq \delta_+.$$

This condition is enforced in Line 8 of Alg. 1. Conditioning on a sufficiently solved anchor enables progressively finer task proposals, down to resolution $\epsilon_m^U \lesssim L^\star \overline{\epsilon}_T^f$.

Max depth $\overline{m}_T$ is set as:

Lemma C.5 (Reachability)**.** *Under Theorem C.3 and Condition C.4, assume the batch size J is a constant. Set*

$$\overline{m}_T := \left\lfloor \frac{1}{2}\log_2\left(\frac{T}{A_T(\log(eT))^2}\right)\right\rfloor_+,$$

where $A_T := \left(\frac{2(c_u+1)\overline{L}\sqrt{C_\lambda \beta_T \gamma_T}}{c_g \epsilon_0^U}\right)^2$, and $\lfloor x \rfloor_+ = \max\{0, \lfloor x \rfloor\}$. Then, under Condition A.3 and Algorithm 1, the optimal level $m^\star$ is introduced by time T.

Meta regret bound. For brevity, we set $\beta_T := \max_{i \in \mathcal{I}_T} \beta_T^{(i)}$ and $\gamma_T := \max_{i \in \mathcal{I}_T} \gamma_T^{(i)}$.

Theorem C.6 (Regret bound)**.** *Under Theorem C.3 and Lemma C.5, let $R_T^\star := L^\star \sum_{s=1}^T \overline{\epsilon}_s^f$ be the uniform single-task regret bound. Let $N_T := |\mathcal{I}_T|$ be the number of instantiated tasks by time T. Then Algorithm 1 satisfies with $C > 0$ w.p. $\geq 1 - (\delta_f + \delta_u + \delta_-)$, such that*

$$\mathcal{R}_T \leq R_T := C\left(1 + \sqrt{N_T}\right)\frac{\overline{L}}{L^\star} R_T^\star. \tag{A4}$$

By $N_T \leq 1 + J(\overline{m}_T + 1)$, and $\overline{m}_T = \tilde{O}(\log T)$, we have $R_T/R_T^\star = \tilde{O}\big(\sqrt{\log T}\big)$ when $\overline{L} = \Theta(L^\star)$.

Proof Sketch. Let $U_t^{\max} := \max_{i \in \mathcal{I}_t} U^{(i)}$. On the high-probability envelope and generation-success events, the instantaneous regret can be decomposed to:

$$\underbrace{(U^\star - U_t^{\max})}_{\text{generation}} + \underbrace{(U_t^{\max} - U^{(i_t)})}_{\text{selection}} + \underbrace{\big(U^{(i_t)} - u^{(i_t)}(\overline{y}_t^{(i_t)})\big)}_{\text{within-task}}$$

(a) *Generation:* $U^\star - U_t^{\max} \leq \epsilon_{m_t \wedge m^\star}^U$, so the sum is $O(c_\epsilon R_T^\star)$ (Lemma C.5 + Condition A.3).

(b) *Selection:* meta-UCB implies $U_t^{\max} - U^{(i_t)} \leq w_{t-1}^{(i_t)} \lesssim \overline{L}\,\epsilon^f_{s_{t-1}^{(i_t)}}$, giving $\tilde{O}((\overline{L}/L^\star)\sqrt{N_T}\,R_T^\star)$.

(c) *Within-task:* Lipschitzness + GP-UCB bound give $U^{(i)} - u^{(i)}(\overline{y}_s^{(i)}) \leq L^\star \epsilon_s^f$, giving $\tilde{O}(\sqrt{N_T}\,R_T^\star)$.

Combining yields (A4). □

3.2 Discussion of the theoretical results

Q. What if we use an oracle task as the comparator? Let $\underline{R}_T := L^\star \sum_{s=1}^T \epsilon_s^{f,(i^\star_{\epsilon^U})}$. For the RBF kernel, the extra overhead is $\kappa_T := R_T^\star/\underline{R}_T = \tilde{O}\big((\log T)^{\overline{d} - d^{(i^\star_{\epsilon^U})}}\big)$, where $\overline{d} := \max_{i \in \mathcal{I}_T} d^{(i)}$.

Q. Isn't $\overline{L}$ hard to estimate? We adapt $\overline{L}$ online via regret-balancing [12], which incurs only a logarithmic overhead; the same idea applies to other hyperparameters. In experiments, GP hyperparameters are fit by marginal likelihood, while remaining constants are fixed ($\epsilon_0^U = 1$, $c_g = 0.5$).

Q. Isn't Condition C.4 strong? Yes—without Condition C.4, the first term in our regret decomposition is irreducible, so sublinear meta-regret is information-theoretically impossible. Empirically, evolutionary LLMs perform well on code and proof generation [3, 7, 13]; our condition makes this commonly implicit assumption explicit. We provide a Bayesian in-context learning analysis [14, 15]: conditioned on mutation history, the posterior mass on the correct refinement type concentrates exponentially fast in the number of in-context examples. We give $\delta_+ \gtrsim \rho_m(\epsilon_m^U) - \epsilon_{\text{ICL}}$, where $\rho_m(\epsilon_m^U)$ is the problem-specific (LLM-independent) success probability of the underlying (*true*) refinement and ϵ_{ICL} is the Bayes approximation error, controlled by pretraining scale and typically small. This supports a nonzero success probability for *refinable* problems with *informative* mutation histories.

3.3 Related Work

Meta BO. Meta-BO either transfers across related tasks [16, 17, 18, 19] or adapts hyperparameters online [20, 21, 12, 22]. Instead, we *meta-optimize* the task.

LLMs + BO works. LLMs have been used to generate hypotheses [23, 24] and code and proofs [25, 26]. In BO, they are applied as surrogates [27, 28, 29], collaborative agents [30, 31] or kernel generators [32] but under a fixed task.

Other BO. Prior work on unknown search spaces [33, 34] and objective selection [35, 36] assumes a fixed task, an assumption violated in our open-ended setting.